AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application.

Listing of Claims:

What is claimed is:

1. (Currently amended)   A method of managing a [[A]] high performance relational database management system, leveraging the functionality of a high speed communications network, comprising the steps of:

(i) receiving collected data objects from at least one data collection node using at least one performance monitoring server computer whereby , and creating a distributed database is created;

(ii) partitioning the distributed database into data hunks using a histogram routine running on at least one performance monitoring server computer such that the data hunk substantially has a target partition size;

(iii) importing the data hunks into a plurality of delegated database engine instances located on at least one performance monitoring server computer so as to parallel process the data hunks whereby and generate processed data is generated; and

(iv) accessing the processed data using at least one performance client computer to monitor data object performance.

2. (Currently amended)   The system A method according to claim 1, wherein at least one database engine instance is located on the performance monitoring server computers on a ratio of one engine instance to one central processing unit whereby the total number of engine instances is at least two so as to enable the parallel processing of the distributed database.

3. (Currently amended)   ~~The system~~ A method according to claim 2, wherein at least one database engine instance is used to maintain a versioned master vector table.

4. (Currently amended)   ~~The system~~ A method according to claim 3, wherein the versioned master vector table generates [[a]] the histogram routine used to facilitate the partitioning of the distributed database.

5. (Currently amended)   ~~The system~~ A method according to claim 4, wherein the histogram routine comprises the steps of:

(i) dividing the total number of active object identifiers by the desired number of partitions so as to establish the optimum number of objects per partition;

(ii) generating an n point histogram of desired granularity from the active indices; and

(iii) summing adjacent histogram routine generated values until [[a]] the target partition size is reached but not exceeded.

6. (Currently amended)   ~~The system~~ A method according to claim 1, wherein the performance monitoring server comprises an application programming interface compliant with a standard relational database query language.

7. (Currently amended)   A high performance relational database management system, leveraging the functionality of a high speed communications network, comprising:

~~(i)~~ at least one performance monitoring server computer connected to the network, each including:

a module for receiving network management data objects from at least one data collection node device ~~whereby,~~ and creating a distributed database ~~is created~~;

3

(ii) a histogram routine running on the performance monitoring server computer[[s]] for partitioning the distributed database into data hunks such that the data hunk substantially has a target partition size;

(iii) at least two database engine instances ~~running on the performance monitoring server computers so as to~~ for parallel-processing the data hunks ~~whereby~~ to generate processed data ~~is generated~~; and

(iv) at least one performance monitor client computer ~~connected to the network~~ for accessing the processed data ~~whereby~~ to monitor data object performance ~~is monitored~~.

8. (Original) The system according to claim 7, wherein at least one database engine instance is located on the performance monitoring server computers on a ratio of one engine instance to one central processing unit whereby the total number of engine instances for the system is at least two so as to enable the parallel processing of the distributed database.

9. (Original) The system according to claim 8, wherein at least one database engine instance is used to maintain a versioned master vector table.

10. (Currently amended)   The system according to claim 9, wherein the versioned master vector table generates [[a]] the histogram routine used to facilitate the partitioning of the distributed database.

11. (Currently amended)   The system according to claim 10, wherein the histogram routine comprises the steps of:

(i) dividing the total number of active object identifiers by the desired number of partitions so as to establish the optimum number of objects per partition;

(ii) generating an n point histogram of desired granularity from the active indices; and

(iii) summing adjacent histogram routine generated values until [[a]] the target partition size is reached but not exceeded.

4

12. (Currently amended) The system according to claim 7, wherein the performance monitor~~ing~~ server comprises an application programming interface compliant with a standard relational database query language.

13. (Currently amended) The system according to claim 7, wherein at least one performance monitor client computer is connected to the network so as to communicate remotely with the performance monitor~~ing~~ server computers.

14. (Currently amended) A ~~storage medium readable by an install server computer in~~ computer program product for a high performance relational database ~~management system including the install server~~, leveraging the functionality of a high speed communications network, the ~~storage medium encoding a computer process~~ computer program product comprising:
     ~~(i) a processing portion for~~ receiving collected data objects from at least one data collection node using at least one performance monitoring computer ~~whereby~~, and creating a distributed database ~~is created~~;
     ~~(ii) a processing portion for~~ partitioning the distributed database into data hunks using a histogram routine running on at least one performance monitoring server computer such that the data hunk substantially has a target partition size;
     ~~(iii) a processing portion for~~ importing the data hunks into a plurality of delegated database engine instances located on at least one performance monitoring server computer so as to parallel process the data hunks ~~whereby~~ and generate processed data ~~is generated~~; and
     ~~(iv) a processing portion for~~ accessing the processed data using at least one performance client computer to monitor data object performance.

15. (Currently amended) The ~~system~~ computer program product according to claim 14, wherein at least one database engine instance is located on the data processor server computers on a ratio of one engine instance to one central processing unit whereby the total number of engine instances is at least two so as to enable the parallel processing of the distributed database.

16. (Currently amended)   The ~~system~~ computer program product according to claim 15, wherein one of the database engine instances is designated as a prime database engine instance used to maintain a versioned master vector table.

17. (Currently amended)   The ~~system~~ computer program product according to claim 16, wherein the versioned master vector table generates [[a]] the histogram routine used to facilitate the partitioning of the distributed database.

18. (Currently amended)   The ~~system~~ computer program product according to claim 14, wherein the histogram routine comprises the steps of:

   (i) dividing the total number of active object identifiers by the desired number of partitions so as to establish the optimum number of objects per partition;

   (ii) generating an n point histogram of desired granularity from the active indices; and

   (iii) summing adjacent histogram routine generated values until [[a]] the target partition size is reached but not exceeded.

19. (Currently amended)   The ~~system~~ computer program product according to claim 14, wherein the performance monitoring server comprises an application programming interface compliant with a standard relational database query language.